Cyber Security

Front End:

Input Validation:

- **Objective:** Ensure that user input is valid to prevent malicious input.
- **Implementation:**
  - **Email Validation:** Ensure that the email address entered by the user is valid.
  - **Time Validation:** Verify that the entered time is appropriate and falls within acceptable operating hours.
  - **Likelihood and Effect:**
    - **Likelihood:** Moderate – Users might enter invalid data or attempt malicious inputs.
    - **Effect:** High – Invalid data or malicious input can compromise the system’s integrity and functionality. And it costs time for staff to contact the user and correct the wrong input.

Authentication:

- **Objective:** Implement a secure user authentication and authorization mechanism so that only employees with an account and password can access the appointment management system.
- **Implementation:**
  - **Username and Password:** Secure authentication requiring valid credentials.
  - **Likelihood and Effect:**
    - **Likelihood:** Low – With proper security measures, unauthorized access attempts should be rare.
    - **Effect:** Critical – Unauthorized access can lead to data breaches and manipulation.

Secure Data Transfer:

- **Objective:** Ensure that data from the reservation interface is securely uploaded to the database, preventing leakage during transmission.
- **Implementation:**
  - **Encrypted Transmission:** Use HTTPS to encrypt data during transmission.
  - **Likelihood and Effect:**
    - **Likelihood:** Low – Proper encryption reduces the risk of data interception.
    - **Effect:** High – Data leakage during transmission can expose sensitive information.

Back End:

Login Authentication

Purpose: To protect sensitive information in the database and ensure that only authorized users can access critical data.

Implementation Measures:

1. **Username and Password Authentication:**
   - Users must authenticate themselves using a username and password to access the system. This provides the first layer of security for access control.
   - **Likelihood and Effect:**
     - **Likelihood:** Low–strong password policies reduce unauthorized access attempts.
     - **Effect:** Critical – Unauthorized access can compromise sensitive data.
2. **Interceptors:**

    - We have implemented interceptors to check the legitimacy of each request. Interceptors first verify whether the request contains valid authentication credentials before allowing access to resources.
    - **Likelihood and Effect:**
        - **Likelihood:** Moderate – Interceptors help, but can be bypassed if improperly configured.
        - **Effect:** High – Invalid requests can still lead to data breaches.
    - We have implemented interceptors to check the legitimacy of each request. Interceptors first verify whether the request contains valid authentication credentials before allowing access to resources.
3. **JSON Web Token (JWT):**
    - We use JWTs for state management, offering a secure way to verify the identity of requestors. JWTs provide a compact, URL-safe means of transmitting information between network applications.
    - The token contains all necessary information to allow the server to verify user identity and ensure the token has not been altered during transmission.
    - **Likelihood and Effect:**
        - **Likelihood:** Low – JWTs are secure when implemented correctly.
        - **Effect:** Critical – Tampered tokens can lead to unauthorized access.

Security Policies:

- **Regular Security Policy Updates:** We regularly update our security policies and verification mechanisms to address emerging threats and vulnerabilities.
    - **Likelihood and Effect:**
        - **Likelihood:** Moderate – Policies need constant updating.
        - **Effect:** High – Outdated policies can lead to vulnerabilities.
- **Encrypted Storage:** All user passwords are encrypted before storage, using currently considered secure encryption algorithms to enhance security.
    - **Likelihood and Effect:**
        - **Likelihood:** Low–strong encryption methods protect stored data.
        - **Effect:** Critical – Compromised passwords can lead to unauthorized access.

SQL Injection Prevention

Purpose: To safeguard our system from SQL injection threats, which can compromise data integrity and security.

Implementation Measures:

1. **Safe SQL Statement Construction:**
    - Developers are required to use parameterized queries or prepared statements when building SQL commands. This practice ensures that user inputs are handled safely and reduces the risk of SQL injection.
    - **Likelihood and Effect:**
        - **Likelihood:** Low – Using parameterized queries reduces the risk.
        - **Effect:** Critical – SQL injection can lead to data corruption or theft.
2. **Input Validation:**
    - All user inputs are rigorously validated against a set of rules (e.g., data type, format, length) before being processed. This minimizes the chances of malicious data entering our systems and being executed as part of SQL commands.
    - **Likelihood and Effect:**
        - **Likelihood:** Moderate – Requires thorough validation practices.
        - **Effect:** High – Malicious inputs can compromise the database.

Security Practices:

- **Regular Code Reviews:** Conduct regular code reviews focusing on database interactions to ensure that secure coding practices are consistently applied across all backend development.
    - **Likelihood and Effect:**
        - **Likelihood:** Moderate – Regular reviews reduce risks.
        - **Effect:** High – Prevents security breaches due to poor coding practices.